

Patent Application

For

A METHOD FOR ANALYZING THE QUALITY OF
TELECOMMUNICATIONS SWITCH COMMAND TABLES

Inventors

John S. Doleac

William T. Freebersyser

Carolyn M. Jack

GTE Ref. No. 99-881

**A METHOD FOR ANALYZING THE QUALITY OF TELECOMMUNICATIONS
SWITCH COMMAND TABLES****Field of the Invention**

The invention generally relates to systems and methods executed in a computer system,
5 and more particularly to systems and methods for analyzing the quality of command generation
performed automatically in a computer system.

Background of the Invention

There are many applications in which commands may be generated automatically to
perform a certain function. One application includes the generation of electronic commands
10 necessary to provision, modify and delete service with regard to telephone subscription service.
In particular, electronic commands may be switch specific commands where a switch is included
in the Public Switch Telephone Network (PSTN).

In some systems and applications, an existing system is replaced by a more updated
system which generates commands, such as the electronic commands, for execution by various
15 telephone switches. In this particular instance, a technique may be used to verify the correctness
of the new system replacing an existing system with regard to the commands generated by the
new system in an automated and more efficient fashion. However, one of the problems is how to
verify the correctness of the new commands generated by the new system in comparison with

GTE Ref. No. 99-881

those of the existing system. One technique is to perform a manual comparison by having a person examine the commands generated by the existing system and those generated by the new system. If there is a match, such as by string or character-by-character comparison, then it may be concluded that the commands generated by the new system are correct. However, this
5 presents several problems in that having a user manually compare these commands, such as by visual comparison, is often inefficient and may introduce unnecessary human error.

An improvement upon this technique is to provide an automated system for comparing the commands such as by a character or string comparison of the existing system commands with those generated by the new system. However, even with an automated method for comparison,
10 additional problems may exist. For example, the commands generated by the two systems may not be identical. In other words, commands may perform equivalent functions but may not be string character identical. For example, features or parameters for a particular command may be in different orders. Additionally, there may be a command generated by the new system which is equivalent to the command generated and used in an existing system. For example, the new
15 provision existing may generate three commands which perform the same function as a single command in an existing provisioning system or vice versa. Neither one is incorrect, however it is difficult to come up with a technique for performing verification of all of these possibilities.

Thus, what is desired is an efficient and flexible system for verifying the quality of the commands automatically generated in a computer system.

GTE Ref. No. 99-881

Summary of the Invention

In accordance with principles of the invention is a method executed in a computer system for verifying generated commands. A subset of actual commands generated by a first system is determined. Using data tables used by a second system in command generation, it is
5 determined if the second system is capable of generating one or more commands equivalent to a first command included in the subset.

In accordance with another aspect of the invention is a system for verifying generated commands used in a computer system. Included is means for determining a subset of actual commands generated by a first system. Additionally included is means for determining, using
10 data tables used by a second system in command generation, if the second system is capable of generating one or more commands equivalent to a first command included in the subset.

In accordance with yet another aspect of the invention is a system for verifying commands. Included are machine executable instructions for determining a subset of actual commands generated by a first system. Also included are machine executable instructions for
15 determining, using data tables used by a second system in command generation, if the second system is capable of generating one or more commands equivalent to a first command included in the subset.

GTE Ref. No. 99-881

Thus, there is provided an efficient and flexible system for verifying the quality of the commands automatically generated in a computer system.

GTE Ref. No. 99-881

Brief Description of the Drawings

Features and advantages of the present invention will become more apparent from the following detailed description of exemplary embodiments thereof, taking in conjunction with the accompanying drawings, in which:

5 Figure 1 is an example of an embodiment of a system that includes the invention;

 Figure 2 is an example of an embodiment of a record structure as may be included in the telephone subscriber database of Figure 1;

 Figure 3 is an example of a provisioning system that may be included in the system of Figure 1;

10 Figure 4 is an example of an embodiment of the various tables used in automatic program generation in the provisioning system of Figure 3;

 Figure 5 is a flow chart of steps of a method of one embodiment for determining what program to execute to generate a switch command;

15 Figure 6 is an example of an embodiment of a data structure that represents a work order as generated in the system of Figure 1;

 Figure 7 is an example of an embodiment of a flow chart including method steps that may be performed in processing a work order;

 Figure 8 is an example of a embodiment of a flow chart including method steps that may be performed by a service request processor in the system of Figure 3;

GTE Ref. No. 99-881

Figure 9 is an example of an embodiment of a flow chart including method steps that may be performed by a network element process included in an embodiment of a provisioning system;

Figures 10 and 11 are examples of embodiment of the various service or work order types;

5 Figure 12 is an example of an embodiment of data identifying service types, command types and associated programs;

Figure 13 is an example of an embodiment of a program for generating a command switch as identified in Figure 12;

10 Figure 14 is an example of an embodiment of a generalized record or line of program of Figure 13;

Figure 15 is an example of an embodiment of another program for generating a command switch;

Figure 16 is an example of an embodiment of a flow chart of method steps for determining a service name;

15 Figures 17 through 20 are examples of embodiments of tables defining a data schema used in generating commands in the system of Figure 1;

Figure 21 is an example of an embodiment of a data structure that includes a portion used for communicating data in the system of Figure 1;

20 Figure 22 is an example of an embodiment of a system used to analyze the commands generated by a new second provisioning system in accordance with log files generated by an existing first provisioning system;

GTE Ref. No. 99-881

Figure 23 is a flowchart of an example of an embodiment of a method including steps for analyzing the commands generated by a new provisioning system;

Figure 24 is a flowchart of an example of an embodiment of method steps of a second method for analyzing commands generated by a provisioning system;

5 Figures 25 through 28 are an example of an embodiment of various reports generated by the analysis program and the embodiment of Figure 22;

Figure 29 is an example of an embodiment of block diagram of a system used to automatically generate switch specific command options based on a first existing provisioning system to be used in a second provisioning system;

10 Figures 30 and 31 are flowcharts of a method of steps of an embodiment used to automatically determine various switch specific commands and command options from an existing provisioning system to be included in use in a new second provisioning system; and

Figure 32 is an example of an embodiment of the order identifier table included in the system of Figure 29.

GTE Ref. No. 99-881

Detailed Description of Preferred Embodiments

Referring now to Figure 1, shown is an example of an embodiment of a system used to generate electronic commands to provision and change service in the PSTN. Generally, the system 10 of Figure 1 includes components of a system as well as data flow inputs and outputs with regard to the various system components. Customer contact data 12 is input into an order and inventory system 14. The order and inventory system 14 obtains necessary telephone subscriber database information from the telephone subscriber database 16 to produce a work order. In this embodiment, this work order is in the form of an order data structure 18 produced by the order and inventory system 14. The order data structure 18 is used as input to one or more provisioning systems 20 and 22. Generally, the provisioning systems take as input the work order and output the necessary commands to the front end server 24 to modify subscriber information in the PSTN 26.

In the system 10 of Figure 1, shown are two provisioning systems, provisioning system 1, 20 and provisioning system 2, 22. In this particular embodiment, the provisioning system 1 may be an existing system in the process of being replaced by a second provisioning system 22. In this particular embodiment of Figure 1, the two provisioning systems may be used alternately depending on which system has been selected for use for operation in the system 10 of Figure 1. For example, if there is a problem experienced in testing and putting the provisioning system 2 on-line, the provisioning system 1 may be selected for operation rather than the new provisioning

GTE Ref. No. 99-881

system 2. Thus, the embodiment of Figure 1 is one particular embodiment in which a new system may be in operation as a phase-in process for replacing an existing system such as element 20. Each of the provisioning systems as will be described in the paragraphs that follow, output commands for particular switches which arrive at the front end server 24.

5 In this embodiment, the front end server 24 is generally a computer system which acts as an interface or a front end to the public switch telephone network or PSTN 26. Generally, the PSTN 26 includes a plurality of switching circuits 28, 30, 32 and 34. It should be noted that other embodiments may include a different number of switches. Generally, the PSTN is given instructions or programmed by commands as passed through the front end server 24 regarding
10 service options for a particular phone number, or to set particular service or subscriber options. For example, as will be described in paragraphs that follow, various commands for a new subscriber for a telephone service may be formulated in the form of a work order as an order data structure 18 produced by the order and inventory system 14. A new telephone subscriber information may be input through customer contact data 12 as by an agent through manual
15 operation and input to the order inventory system 14. In turn, the order and inventory system may produce and record information in the telephone subscriber database 16 with the new subscriber's telephone number. To begin service on a particular date, for example, a work order is produced in the form of an order data structure 18. This order data structure 18 is an input to one or both of the provisioning systems in operation which output the switch commands for this
20 particular subscriber to enable service on a particular date. These commands are forwarded to

GTE Ref. No. 99-881

the front end server 24 which then passes these commands to program one of the switches as may be included in the PSTN 26 for the particular subscriber.

Generally, the provisioning systems, such as provisioning system 1 and 2 of the system 10 of Figure 1, are a collection of software programs executing on a computer system that together generate electronic commands to provision, modify, and delete service in different types of stored program controlled telephone switches as may be included in the PSTN 26 of Figure 1. For example, the switches that may be included in the PSTN 26 include an AT&T GTD-5, Nortel DMS-100, Nortel DMS-10, AT&T 5ESS, Vidar, Siemens DCO and Siemens EWSD switches.

It should be noted, as will be described in paragraphs that follow, that different commands are generated by each of the provisioning systems in accordance with the different types of switches that may exist in the PSTN. Also, in the embodiment of Figure 1 in this particular example, the provisioning system 1, 20 is an existing system to be replaced by a more updated flexible system, provisioning system 2, 22. Provisioning system 1 in this particular embodiment includes existing computer programs which are not written in modular form and are difficult to maintain. For example, the existing code included in the provisioning system 1, 20 includes knowledge about specific requirements for different switch types imbedded in the code. As a result, maintenance of this code is difficult as well as adding new switches and code for these switches into the existing system with the particular coding model used which is non-

GTE Ref. No. 99-881

modular. Thus, it is being replaced by a second provisioning system 22 which is deemed to be more efficient and flexible. However, phasing in the real time system provisioning system number 2 is needed such that if a problem is encountered with the second positioning system, a switch over to the use of the first or existing provisioning system 20 is performed. Thus, this is the reason for the two provisioning systems included in this particular example and the embodiment of the system of Figure 10.

It should be noted that other embodiments and applications may include more provisioning systems or less provisioning systems. For example, once the first provisioning system has been deemed obsolete and a second provisioning system is been deemed thorough tested, the first provisioning system 20 in Figure 1 is removed and the order data structure 18 serves as an input only to provisioning system 2, 22. As will be described in paragraphs that follow, the second provisioning system 22 generates a variety of commands for a variety of different switch types as previously described. The new and improved provisioning system 22 includes modular code which provides for flexible system maintenance and reusable code.

In this embodiment, the switches that may be included in the PSTN are stored program control switches. Generally as known to those skilled in the art, each of these switches has a memory which may vary in size which has it own particular command set. Generally, a command is forwarded from the front end server 24 to each of the switches as directed for a particular telephone subscriber. The various switches each respond back to the front end server

GTE Ref. No. 99-881

with regard to the status of the command completion. For example, the status may be "success" or "failure". This status is further forwarded from the front end server 24 to the provisioning systems 20 and 22 as will be described in paragraphs that follow.

Generally, the front end server 24 acts as an interface between the provisioning system
5 and the PSTN. The front end server 24 in this particular embodiment includes support for the TCP/IP protocol and translates any incoming commands for use in accordance with the X.25 standard protocol as understood by those switches included in the PSTN 26. Thus, the front end server 24 acts as an interface between the TCP/IP standard and the X.25 standard.

The provisioning system 22 as will be described in paragraphs that follow in this
10 particular embodiment includes use of the Automatic Service Activation Program™ (ASAP™) application which is generally an infrastructure product produced by Architel Systems Corporation™. It should be noted that other systems may be used in other embodiments to implement systems in accordance with the principals of the invention as will be described herein.

In this embodiment, the provisioning systems, such as provisioning system 1 and
15 provisioning system 2 included in the system 10 of Figure 1, are executed on a computer system that has the HP/UX operating system with a Sybase database management system. The ASAP™ system by Architel™ is included in the provisioning system 22.

GTE Ref. No. 99-881

It should be noted that each of the provisioning system and the other components included in the system 10 of Figure 1 may be associated with various hardware configurations. In one embodiment, for example, the order and inventory system 14, each of the provisioning systems and the front end server 24 each execute on dedicated processors in different computer
5 systems. However, as known to one of ordinary skill in the art, other hardware configurations are possible in accordance with principles of the invention as described herein.

It should be noted that in one particular embodiment, for example, the order and inventory system 14 and the telephone subscriber database 16 each run on a commercially available IBM mainframes. Provisioning system 1 may execute on a single computer system or
10 a combination of commercially available processors that together form the provisioning system 1. Provisioning system 2 may be a commercially available processor from Hewlett Packard, such as the K460, running HP UX Unix. The front end server 24 may be a commercially available processor running IBM AIX.

Referring now to Figure 2, shown is an example of data structures in the form of tables as
15 may be included in the telephone subscriber database 16 of Figure 1. Generally, shown in Figure 2 are two tables 36 and 48. The first table 36 includes telephone number and feature information for a subscriber. The second table 48 includes address information. In this particular embodiment, the telephone number of record 38 may be used as an index into the table 36 for an existing customer. Associated with a particular telephone number 44 are various features and

GTE Ref. No. 99-881

other information 40. Additionally, associated with a particular telephone number 44 for a particular record is a pointer or identifier 42 to address information 46 for a particular subscriber. The address information in this particular embodiment is organized in a second separate table 48.

5 This particular type of data arrangement may be used in the order and inventory system 14 of Figure 1, for example, when the customer contact data input into the order and inventory system may include information for a new or existing subscriber for a telephone service. For example, if there is information being input into the order and inventory system for a change order for an existing telephone subscriber, a telephone number exists already for that subscriber and information may be obtained and stored from these tables accordingly by indexing into table 10 36 by telephone number. However, if a customer or subscriber is new and does not have an existing telephone number in the database, table 36 may not initially be used. Thus, information is input into table 48 including information about the address of the subscriber. Subsequently, the order and inventory system 14 determines a telephone number which is available by looking into the telephone subscriber database. For a new subscriber the process works in reverse in that 15 the address information may be input first. A telephone number may be determined which is available and inserted into the table 36. Subsequently, this newly created telephone number and information is associated with the address information in table 48.

It should be noted that other embodiments of the telephone subscriber database and other information may be included in the database. This is one particular organization of telephone

GTE Ref. No. 99-881

subscriber information. It should also be noted that other embodiments of the system 10 of Figure 1 may include multiple order and inventory systems, for example, where there are multiple order data structures output into a single provisioning system as well as multiple provisioning systems.

5 Referring now to Figure 3, shown is an example of a provisioning system 22 as may be included in the system of Figure 1. It should be noted that the provisioning system 22 is an example of an embodiment of the newer provisioning system 22 which is being phased in to replace an older existing system 20. Input into the provisioning system 22 is the order data structure 18 produced by the order and inventory system 14 previously shown in Figure 1.

10 In one embodiment, the input order data structure 18 may be an existing order data structure that is put in a new and improved format as used within the provisioning system 2. For example, the order data structure 18 may be a fixed length record with a fixed format. In one embodiment, this data structure 18 may include over 100 fields including record type, order number, switch number, telephone number, class identifier indicating feature data, transaction
15 type and the like. Data included in different fields may have different meanings based on values included in other fields. Other embodiments may include different formats and fields for the order data structure 18 in accordance with the features and applications of the other embodiments.

GTE Ref. No. 99-881

The order data structure 18 is input to the service request processor (SRP). Generally, the SRP receives orders, translates or reformats them in accordance with another data structure, assigns an order type to the work order, and enters information into the service order database 54. Generally, as will be described in more detail the order types are defined as common service
5 description layers or CSDLs. The SRP 50 interacts with the task coordinator 52. Generally, the SRP stores the orders in the database. The task coordinator 52 selects the orders according to due date and status. The task coordinator 52 communicates to the SRP when an order is completed and the corresponding final status. Subsequently, the SRP communicates this information to the order and inventory system. This processing is described in more detail in
10 paragraphs that follow.

It should also be noted that in this embodiment, the SRP and the task coordinator are daemons that execute independent of each other on a computer system of the provisioning system 2. Other embodiments may implement each of these using various techniques known to those skilled in the art and in accordance with each embodiment.

15 The task coordinator 52 queries or retrieves information from the service order database as needed in accordance with processing various service or work orders. For example, the task coordinator 52 may retrieve records from the service order database 54 with regard to a particular due date and all of the work orders which must be processed by a particular date and/or time.

GTE Ref. No. 99-881

Generally, the task coordinator 52 serves as a task allocator and coordinator of the various work orders to be processed. The task coordinator 52 determines which network element process (NEP) produces the various commands corresponding to a particular order. In this particular embodiment, the task coordinator determines which NEP processes a particular order
5 in accordance with a switch identifier, or switch id. In this particular embodiment, the task coordinator may read a switch id mapping into an NEP process table from the service order database 54. Generally, this table maps a particular switch identifier or group of identifiers to a particular NEP process. At start up, the task coordinator may read this table from the service order database so that it knows how to partition the various work orders.

10 It should be noted that a switch identifier is a unique identifier of one of the switches included in the PSTN 26. Each switch identifier uniquely identifies a physical switch included in the PSTN 26 of Figure 1. In this particular embodiment, it should be noted that the task coordinator 52, the SRP 50, and each of the NEPs 56 through 60 may be written in the C programming language. In this particular embodiment they are included as part of the ASAP™
15 package or software produced by Architel Systems Corporation™. In this particular embodiment, the SRP may be modified in accordance with the particular data structure, for example, as well as other system particularities which may vary with each embodiment. The NEP and task coordinator are commercially available programs included with the ASAP™ applications which are not generally modified. However, other embodiments may provide

GTE Ref. No. 99-881

similar function as will be described in this application to perform the functions in accordance with principles of the invention.

In this particular embodiment, there are 25 NEP processes. Other embodiments may include more or less number of NEP processes as well as determine which processes perform a process the various orders as determined by the task coordinator. Generally, a load balancing technique is incorporated into the task coordinator 52 by using a switch identifier as an index into NEP process tables read in at task coordinator start up. These tables, as previously described, perform the function in this embodiment of load balancing the various NEP processes to process the service or work orders included in the service order database 54. This table may be modified in accordance with the various work order or switch ID mappings with the goal of having efficient system performance.

In this particular embodiment, it should be noted that the switch identifier which is used as an index to a table to determine which NEP process processes this particular work order is an eleven character unique identifier. Also, in this embodiment, the load balancing and the division of which switch identifier is associated with a particular process for processing a work order is based on historical data gathered from execution of the system 10 of Figure 1. In other words, in this particular embodiment it is determined that based on past performance and statistics gathered at various points in the system, particular switch identifiers have an associated throughput or load of a number of work orders. Accordingly, the system is load balanced for current use based on

GTE Ref. No. 99-881

this past data gathering. Other embodiments may use other techniques for determining this mapping.

In this embodiment NEP 56 through 60 are identically coded modules. The task coordinator 52, in its initialization, may also input a CSDL- ASDL table. This CSDL-ASDL mapping table lists for each particular CSDL or work order type, one or more associated command types or ASDLs to be associated with a particular work order type. For example, if an order is for an initialization of service, a CSDL is associated with this work or service order type representing this initialization of service. Accordingly, this CSDL or service order type is associated with one or more ASDLs or command types to accomplish the type of service associated with a particular service order type. It is one of the functions of the task coordinator to read the CSDL-ASDL tables on start up and determine for a particular order retrieved from the service order database 54 what ASDL commands are needed to carry out the operations associated with a particular service or work order type. Generally, each CSDL may be broken down into sub processing steps, each of the sub processing steps may be associated with an ASDL.

Subsequently, for each ASDL command, the task coordinator interacts with the associated NEP process to execute the appropriate command for the particular switch. The task coordinator sends a message to the NEP process which includes the ASDL command, the switch identifier associated with a particular work order, and other data necessary to carry out the ASDL

GTE Ref. No. 99-881

command as may be identified for example in the work order. The NEP process then retrieves the program from the NEP database 62 and executes the appropriate program in accordance with the ASDL command for a particular switch to generate the commands which are sent to the front end server to the particular switch as identified in this system 10 of Figure 1.

5 Referring now to Figure 4, shown is an example of an embodiment of how the various CSDL-ASDL and program tables interact to facilitate the generation of the various switch commands. It should be noted that the various tables identified in Figure 4 may be used by the task coordinator and NEP process to determine which program to execute. Shown in Figure 4 is a CSDL table 70, an ASDL table 72, and a program table 74. The CSDL table 70 includes
10 various service order types as will be described in paragraphs that follow. The CSDL 70 maps service order types to one or more ASDL commands or command types as may be included in the ASDL table 72. In this particular example shown in Figure 4, the service order type 76 requires the execution of two command types 78 and 80 as included in the ASDL table 72 to fully implement the service order type 76. The first step in generating the commands necessary
15 to carry out the service order type 76 are those associated with command type 78. Recall that the task coordinator determines the CSDL to ASDL mappings. Thus the task coordinator retrieves and stores the information as may be included in the CSDL and ASDL tables describing what ASDLs are associated with what particular CSDLs.

GTE Ref. No. 99-881

In this particular embodiment, for example, these tables and information may be stored in a service order database 54. Information is sent to the appropriate NEP process in accordance with the previously described load balancing tables also read by the task coordinator for the service order database 54. Once the NEP process has been transmitted the correct information with regard to ASDL, the NEP then maps a particular ASDL to a particular program. This precise mapping, such as for a particular ASDL 78 mapping to a particular program 82 as included in the program table 74, will be described in more detail in paragraphs that follow using various pieces of data and other information. Generally, the program table 74 in this particular embodiment may be implemented as records stored in the NEP database 62.

The model used in the table mappings provided by Figure 4 associates one service order with one or more command types which must be performed in combination to accomplish generating the commands associated with a particular service order type. The mapping from a particular command type, such as 78, to a particular program, such as 82, maps a single command type to a single program as may be included in the program table 74. In other words, for each command type such as 78, one program is selected from the program table 74 to be executed. It should be noted that each command type, such as command type-1 78, may be associated with one or more of the programs, such as 78 and 79, included in the program table. However, to be distinguished from the CSDL to ASDL mappings, only one program is executed for a particular ASDL in this embodiment. This is in contrast to the CSDL, ASDL mapping where more than one ASDL command may be executed sequentially to accomplish generating

GTE Ref. No. 99-881

the commands for a particular service order. It should be noted that in this embodiment, the ASDL to program mapping tables may be stored in the NEP database 62 since it is used and accessed by the NEP processes. Other embodiments may include other data storage and management configurations.

5

Referring now to Figure 5, shown is an example of an embodiment of the steps included in a flow chart as may be performed by the NEP process in selection of a particular program in reporting the status to the task coordinator. At step 90, the NEP uses the switch identifier to obtain the software level and software type associated with a particular switch identifier.

10 Generally, this information may be included in a tabular format stored in the NEP database 62 indexed by switch identifier as a key that returns the software level and the software type of a particular switch identifier. At step 92, using the ASDL as sent in the message by the task coordinator, the switch type and software level as determined by the look up in the table in NEP database 62 from step 90, a program is selected for execution. Similarly, these program tables as
15 indexed by ASDL, switch type and software level may be stored in the NEP database 62. In other words, using as a key the ASDL, the switch type and the software level, a particular program name is selected and returned as a result of output from a database query 62.

At step 94, a designated NEP process executes the program as may be stored in the NEP database 62. At step 96, upon subsequent program execution, the NEP process receives the

GTE Ref. No. 99-881

status after the execution of the program and the command by the switch, and reports status results to the task coordinator 52.

It should be noted that the various programs previously described executed by the NEP process are stored in this embodiment in the NEP database 62 as a variety of records. An
5 example of the various programs executed by the NEP process are described in more detail in paragraphs that follow.

It should be noted that as previously described only one program of one or more programs is executed for each ASDL or command type defined in the ASDL table 72 in this embodiment. Each program as may be included or identified in the program table 74 generates a
10 different command in accordance with the switch type and software level. One aspect of the technique incorporated in the tables of Figure 4 is to reduce the number of CSDLs or service order types by embedding software type dependencies in the ASDL and program choosing process. Thus, each of the order type or service orders as included in the CSDL table 70 are independent of switch type. Generally, in this embodiment, a switch type is associated with a
15 particular type of switch as may be included in the PSTN. This switch type is generally a vendor categorization about a particular switch identifier in accordance with, for example, the make and model of a particular switch. Thus, what is described herein is a unique way for choosing the various service order types and ASDL organizations and the mappings between the CSDL, ASDLs, and the various programs executed. Using the various mapping techniques described

GTE Ref. No. 99-881

herein, ASDLs may be reused in up to more than one CSDL. Similarly, the code associated with each of the programs included in the program table may be executed more than one time by being associated with more than one ASDL. This reuse of code as well as the fact that each ASDL may be mapped or associated with more than one CSDL may result in decreasing storage space by reuse of the commands and associated programs. This provides for efficient use of data storage and may increase the maintainability of a particular system by having one copy of a particular code segment which may be reused several times. Thus, when a particular program segment requires modification, this may generally be performed on a single copy of the code. In summary, the multiple level mappings and hierarchy as set forth in the tables in Figure 4 provide for code reuse, facilitate modular program design, and facilitate a flexible and maintainable system.

Referring now to Figure 6, shown is an example of an embodiment of the data structure that may be initialized by the SRP 50 and stored in the service order database 54. The data structure 100 includes several different types of information. Generally, some of this information may be obtained from the order data structure 18 which is input into the SRP 50. Other types of information are filled in by the SRP by looking up information from the service order database 54. In this particular embodiment, the data structure 100 includes a service order ID 102. The service order ID uniquely identifies this particular work order. The service order ID is obtained from the order data structure 18 in this particular embodiment. The service order

GTE Ref. No. 99-881

type 104 or CSDL is determined by the SRP. In this particular embodiment, the SRP determines the service order type by accessing an order type table.

It should be generally noted that tables, such as the order type table, that will be described in conjunction with Figure 6 will be described in more detail in accordance with other Figures in paragraphs that follow. An embodiment of the order type table is described in conjunction with other Figures. However, note that the service order type or CSDL is determined by performing a look up in accordance with the table stored in the service order database 54 of Figure 3 by using information obtained from the order data structure 18 as an index to obtain the order type or CSDL included in field 104. The telephone number 106 associated with a particular work order is obtained from the input order data structure 18. Similarly, the switch identifier 108, due date 110, user priority level 114, various features identified as parameter values 118 and 120 are also obtained from the order data structure 18. The switch identifier 108 uniquely identifies a particular switch within the PSTN 26 for this particular subscriber. The due date 110 identifies the date by which this order must be processed. The voice mail server 112 is obtained by the SRP if voice mail is an option for this particular user by working in the voice mail server identifier tables. A detailed description of this particular table will be described in conjunction with other Figures. User priority level 114 may be input from the original order data structure identifying the particular priority of this work order request. For example, if a user or subscriber has service that has been affected by a disaster, such as a hurricane or flood with only partial service restoration at first, the user priority level maybe elevated as indicated by field 114 of the

GTE Ref. No. 99-881

data structure to determine the priority level as to which subscribers' service will be restored first, such as restoring service to a hospital or police station prior to a resident. The status 116 indicates the status of this work order as, for example, whether it has been processed successfully or failed, or whether it has yet to be completed. Other features associated with a particular work order request 18 may appear in fields 118 through 120.

It should generally be noted that features such as due date, voice mail server, user priority level, status and others as may be associated with the work order data structure 18 are parameter-value pairs in this embodiment. In other words, they are pairs of information where a first parameter may be an alphanumeric string identifying a particular feature, and the second parameter is a value, such as a numeric value or a character or string, associated with or giving an instantiation of a particular feature.

It should be noted that what has just been described in conjunction with Figure 6 for data structure 100 is one particular embodiment of the data structure as may be produced by the SRP and stored in the service order database. Other embodiments may include different data structure organizations in accordance with the various features of that particular system. Additionally, one feature of this data structure is the parameter value pairs to describe the various features. Another type of general data organization to describe the different features or items associated with a work order data structure may be used in other embodiments.

GTE Ref. No. 99-881

Referring now to Figure 7, shown is a flowchart of the method steps of tasks performed in a provisioning systems to produce the various switch commands in accordance with a particular switch type and a subscriber's work order. At step 150, the SRP reads the input work order and information from tables included in the service order database to produce a service order which is stored in the service order database. Generally, the SRP uses information
5 incoming in the data structure 18 and various tables as stored in the service order database 54 to produce a work order which is stored and used further in the provisioning systems. This work order in this embodiment is stored in the service order database 54.

At step 152, the task coordinator retrieves the service order as input to the service order
10 database by the SRP 50. The task coordinator retrieves service orders from the service order database to process orders with a particular due date and status. It should be noted that in other embodiments, other criteria may be used by the task coordinator to retrieve service orders. In this particular embodiment, the task coordinator retrieves service orders in accordance with due dates and status of initialized or not completed yet. Thus, these orders are retrieved by the task
15 coordinator from the service order database 54 to be completed by generating switch commands.

At step 154, the task coordinator determines which one or more ASDLs or commands types are associated with each CSDL order type. Recall that a CSDL service or work order type is a particular type of order such as a new service order, a modification to a subscription of telephone service, or deletion of a subscription. Other systems may include other order types.

GTE Ref. No. 99-881

For each of these order types, one or more ASDL or command types are associated with each of these CSDLs. For example in order to modify a particular service having five parameters or feaures, there may be 5 ASDLs associated with this CSDL. Each ASDL is associated with producing one switch command which is executed by the switch. Thus, if one CSDL may be
5 further divided into one or more atomic units, each of these one or more atomic units correspond to a particular program executed to produce a single command for the switch.

At step 156, the task coordinator determines which NEP handles processing of the ASDLs associated with each CSDL in step 154. For each ASDL, the steps 158 and 160 are executed. At step 158, a message is sent to the appropriate NEP with the ASDL and the other
10 information needed to execute a program producing and sending corresponding switch commands to the front-end server. Generally, the task coordinator produces a message including necessary information for the NEP to produce select a particular program in accordance with a particular switch type and information to produce a switch command and equipped with a particular work order and switch type.

At step 160, a status is received from the NEP process regarding the status of command execution. Accordingly, the task coordinator updates the status field of the service order data structure to be described in paragraphs that follow. It should be noted that the status information and other pieces of feature information may be included in the service order data structure stored
20 in service order database 54. In this particular embodiment, the status information is stored as a

GTE Ref. No. 99-881

feature value pair with "STATUS" indicating the field or feature name, and a status value accordingly being an alphanumeric or numeric value, indicating the status of the current associated service order. For example, the status of a service order may be "completed" and a numeric order or alphanumeric string may be used to indicate this value. It should be noted that other types of values may also indicate the status value stored in the service order data structure in the database 54. Steps 158 and 160 are repeated for each ASDL associated with the CSDL order type as determined in Step 154. Upon completion, the task coordinator stops processing associated with this particular work order.

Referring now to Figure 8, shown is flowchart of the steps of a method of certain tasks performed by the SRP 50 of the provisioning system of Figure 3. At step 130, the SRP determines the CSDL order type. The SRP determines the order type using subscriber information obtained from the order in the structure 18 from the order type table which will be described in paragraphs that follow as contained in this embodiment in the service order database 54. At step 132, the voice mail server number is determined. This may be determined by the SRP in conjunction with using the voice mail server table as will be described in conjunction with other figures in paragraphs that follow. Similar to the order type table, the voice mail server table and other tables which will be described in conjunction with steps that follow are stored in the service order database 54 of Figure 3 and access by the SRP.

GTE Ref. No. 99-881

At step 134, the line equipment identifier is determined for the particular subscriber. The SRP may determine this by using an equipment translation table which outputs for a particular switch identifier and equipment identified and translated equipment identifier for that particular equipment according to the particular subscriber's hardware and connection. At step 136, using a
5 feature identifier as may be included in the data structure for the work order form 18, a feature name and value as included in the feature value pair of a data structure to be stored in a service order database are determined. The feature name is determined by using the feature name table which will be described in the paragraphs that follow to determine feature name or value.

At step 138, the default features are determined using a default feature table which will
10 also be described in paragraphs that follow. Generally, the default feature may be a default subscriber's feature in accordance with a particular geographic area or other designation. For example, the default feature for a particular state may be dial pulse. A default feature table gives a default feature value for a particular name of a feature. At step 140, a switch identifier is determined using the switch identifier table as will be described in paragraphs that follow as
15 well. Additionally, other information may be obtained by the SRP using tables stored in the service order database 54 to complete an order data structure which will be stored in the service order database.

Generally, one technique used by the SRP is to make use of some values provided in the input data structure from the work order 18 in conjunction with table lookups, as stored in the

GTE Ref. No. 99-881

service order database 54, to fill in fields included in a new order data structure stored in the service order database 54.

Referring now to Figure 9, shown is a flowchart of an embodiment of method steps and tasks performed by the NEP in this embodiment. It should be noted that the steps outlined in this flowchart are only some of those steps which may be performed by the NEP in this allocation of tasks to the NEP. Additional tasks may be performed by the NEP in accordance with other embodiments. At step 162, a message sent by the task coordinator is received by the NEP. In this particular embodiment, the message may include the ASDL command, switch identifier, a telephone number, and any other information dependent on the particular ASDL command.

At step 164, the NEP determines the switch type and associated software level in accordance with a switch ID. In this particular embodiment, this may be performed by the NEP accessing information that may be stored in the table in the NEP database 62. In one embodiment, the NEP database may be an SQL-compliant database which retrieves or returns the switch type and software level when the switch ID is used as an index of a particular table.

At step 166, the NEP determines the program to execute in accordance with the ASDL, the switch type, and the software level. Similar to the technique used in step 164, this information may also be retrieved from the NEP database 62 by performing an SQL query

GTE Ref. No. 99-881

command which returns a particular program name given the ASDL, switch type and software level used as an index. At step 166, the NEP executes this program.

5 In this particular embodiment, the program is executed by the Architel™ software system. Other embodiments may use other techniques in accordance with each specific embodiment and implementation.

In this embodiment, the NEP interprets the instructions associated with each program, and starts a new thread for each order. It should be noted that to process a single order several programs may have to be executed in sequence. Each program may be handled by the same thread which ends execution when the order is complete.

10 At step 168, the program executed in step 166 produces a switch command and sends it to the front-end server 24 for execution by the appropriate switch. It should be noted that in this particular embodiment, a messaging scheme may be used to send the command to the front-end server. This may be in the form, for example, in this embodiment of a TCP/IP message command. At step 170, command execution status is received by the NEP also in the form of a
15 message and returned to the task coordinator accordingly. It should be noted that command execution status is forwarded from the switch to the front-end server, which then further forwards it to the NEP in the form of a message. Command execution status may be, for example,

GTE Ref. No. 99-881

"success" or "failure" indicating the status of the command executed by the switch as associated with the ASDL.

It should be noted that each NEP process in this embodiment using the ASAP™ application is a multi-threaded process written in the "C" programming language. The NEP process executes programs that are written in an interpretive language by Architel™ which generates machine executable code. Other embodiments may employ other techniques in accordance with the software and hardware included in the implementation.

It should be noted that in this particular embodiment, a parameter provided is the maximum number of threads per process. Other embodiments may have other mechanisms for controlling the number of threads associated in a particular process. Additionally, even using the existence of a multi-threaded process is highly dependant upon the underlying software system and capabilities thereof in accordance with each particular embodiment.

It should be noted that each program is written in a scripted interpreted language which is specific to this implementation by Architel™.

Referring now to Figures 10 and 11, shown is a table of various service or work order CSDL types and their associated description in one particular embodiment of the invention. Generally, the CSDL or the work order type is shown in column 180 with a corresponding

GTE Ref. No. 99-881

description in column 182. The various CSDLs included in column 180 may be used, for example, in the field 104 of data structure 100 as included in Figure 6 for the service order types. Generally, these various service or work order types (CSDLs) have been categorized such that they are switch independent. In other words, each of these CSDLs represents a first in a series of hierarchies as shown in Figure 4 with the various ASDLs and programs to produce a data scheme for accessing information in which the CSDLs are switch independent. It is a design decision in this embodiment to embed the switch type dependencies in the hierarchy in levels other than at the top level of the service order type. This is one of the techniques used in this particular embodiment to minimize the impact of adding the difference switch or modifying a particular switch type upon an existing data accesses in the system. This facilitates maintenance as well as the debugging when changes are made to the system as shown in Figure 1.

The various CSDLs or work order types are used to add, change, delete, initialize, link, restore or suspend various telephone services of a particular subscriber. For example, in column 180 number 184 included is a CSDL or work order type for deleting a particular telephone subscriber line service. Record 186 of Figure 11 shows C-SUSPEND_LOCAL which is the corresponding work order or service order type to suspend local service for a single line. This may be used, for example, when an individual has not paid a various portion of their phone bill.

It should be noted that other embodiments may include other service order types in accordance with the various application. It should generally be noted that one feature of

GTE Ref. No. 99-881

particular importance as described is that the CSDLs or work order types are independent of the switch type.

Referring now to Figure 12, shown is an example of an embodiment of the various tables associated with the CSDL C-DELETE_LINE as shown in Figure 10. Referring now to Figure 12, associated with the CSDL or service order type C-DELETE_LINE are four ASDLs or commands 190. Generally, this CSDL deletes a working service. The four ASDLs associated with the CSDL C-DELETE_LINE are each associated with a particular command that must be executed by the switch of the particular subscriber to complete the service order.

It should generally be noted that the format of the information included in Figure 12 is in accordance with the particular specifications of the ASAP™ application as used in this particular embodiment. For each CSDL specified herein, there are three items associated with each ASDL included in the group 190. Column 192, specifies the name of the ASDL. Column 194 contains a single character boolean which is a test performed to determine whether or not the particular ASDL is executed. Column 196 indicates a parameter used in conjunction with the boolean in column 194 to determine if a particular ASDL specified in column 192 is executed. This will be described in more detail in the paragraphs that follow.

Generally, the boolean tests and ASDLs are executed in a linear fashion sequentially from the top to the bottom. In this particular embodiment, the boolean conditions and ASDLs from

GTE Ref. No. 99-881

194 as included in group 190 will be executed in accordance with the CSDL C-DELETE_LINE.

The first ASDL, A-DETERMINE_SR, is executed if a smart ring number (SR_NUMBER) is present (defined as indicated by the "D" boolean condition) on the work order. A-

DETERMINE_SR determines the switch type. The task coordinator determines the one or more

5 ASDLs for a given CSDL by examining the values in column 194 and 196.

Referring to Figure 12, the task coordinator uses tables stored in the database to determine the ASDLs included in area 190 for the CSDL c-delete line. Accordingly the task coordinator determines if the SR_NUMBER 196 is defined as set forth by line one of Figure 12. If the boolean condition indicated by column 194 evaluates to true such that SR number, which
10 stands for smart ring number is defined in the work order data structure, the task coordinator may send a message to the designated NEP process with the appropriate information indicating that the ASDL determine_SR is to be executed.

Control is passed to the NEP process which uses information in the ASDL table 197. The ASDL table 197 includes three pieces of information for each ASDL. In this particular
15 embodiment, the ASDL state table includes an entry for each ASDL as identified in the group 190. For example, the A-DETERMINE_SR ASDL as identified in column 192 has a corresponding entry 198 in the ASDL state table 197. Similarly, A-DELETE_SMART_RING as included in group 190 of the CSDL ASDL table 195 includes an entry 200 in the ASDL state table 197. Associated with each ASDL entry and ASDL state table are 3 columns of

GTE Ref. No. 99-881

information. The first column 204 indicates the switch type. The second column 206 indicates the software level of that particular switch type and the third column 208 indicates the program to be executed. In this particular example, for the ASDL A-DETERMINE_SR for the switch type "DCO" and software level being "REL20", the program determine_SR is executed. It

5 should be noted in this particular embodiment that the determine_SR program as indicated in column 208 is executed for each switch type and is independent of switch type. Thus, in this example, the same program is mapped to the ASDL five times for each of five different switch types. This may be contrasted to the next ASDL in Figure 12 where a different program is mapped to an ASDL for each switch type. Upon the NEP process initiating execution of a
10 program, a status is returned to the NEP process as previously described and control returns to the task coordinator with the status reported. The task coordinator proceeds with executing the subsequent ASDLs in accordance with the defined boolean condition and parameter as indicated by columns 194 and 196 for each of the subsequent ASDLs.

Referring now to Figure 13, shown is an example of an embodiment of the determine_SR
15 program as indicated in column 208 of Figure 12. Generally, the determine_SR program prepares data for determining if smart ring is present. It sets a particular parameter which causes subsequent ASDLs to be executed in accordance with certain parameters and their values.

Generally, the logic performed within the program determine_SR is if the switch type is a DMS-
100 switch, the DMS-100_SR parameter is set which causes the second ASDL as shown in
20 Figure 12 to be executed. Similarly, if the switch is a 5ESS switch, another parameter is set

GTE Ref. No. 99-881

which causes the third ASDL to be executed. For other switch types in this particular embodiment of the determine_SR program it is not necessary to delete the smart ring number explicitly.

Referring now to Figure 14, shown is an example of the generalization of a format for a
5 record as may be included in a program such as determine_SR Figure 13. Generally, a record or a command line included in one of the programs includes a format of 3 fields or three parameters. The first field 210 identifies a specific action or command. Referring back to Figure 13, an example of an action or command is IF_THEN. The second parameter or field 212 is an alphanumeric string. The actual use of the alphanumeric string generally depends on the
10 type of action command in the field 210. For example, if the action or command is IF THEN the alphanumeric string represents a particular test to be performed in terms of boolean logic. The third parameter 214 is an integer value. Similar to the use of the second parameter 212, the integer value 214 and its use are also determined in accordance with the various action or command in field 210.

15 Referring now to Figure 15, shown is an example of an embodiment of a program to initialize a single line service. In this particular embodiment, the program of Figure 15 corresponds to a program associated with the CSDL as indicated in Figure 11, element 185 C-INIT_NON_HUNT_LINE to initialize a single line service. This, for example, may be used to initialize a non-hunt line, such as a user or subscriber telephone service to a residence. As

GTE Ref. No. 99-881

indicated in the comment at line 220, this program is associated with an ASDL that has been determined in accordance with a Vidar switch type.

Referring back to Figure 14, note that each of the lines of the program of Figure 15 conform to the format generally described in accordance with Figure 14, and also as generally followed by A-DETERMINE_SR program in Figure 13. In the next few paragraphs that follow, the logic of the program in Figure 15 will generally be described.

At 222, a call is made to a library function to reformat the line equipment identifier with a specific switch format. In this particular embodiment for this call, it should be noted that "LIB_UTIL" indicates that a general routine exists in this library and is executed for all switch types. This is an indication of how code may also be reused for each particular switch by making a call to a routine. Other routines may be specific to switch types as will be described in paragraphs that follow.

At line 224, the call is made to get a service name. Generally, the classification and representation of the service name is a key component of the data scheme of this system and how the services are organized. Generally, a service name is a codification of a combination of service features that represent a service. As previously described, service features are those features that may be represented by the value parameter pairs. Although the switches included in the PSTN may provide over 400 different features in this particular embodiment, not all of the

GTE Ref. No. 99-881

features may be considered as class-of-service-affecting. Some are known as class features and may be added to a basic service of class-of-service-affecting features requiring the modification of a specific parameter defining the service. One of the reasons for upgrading the provisioning system in this particular embodiment is due to the classification of services in the overall data
5 scheme and design.

The provisioning system 1 in this embodiment generally treats all service features as class-of-service-affecting thus treating all services in a single hierarchy resulting in approximately a million different classes of services each one a different combination of features. The scheme used in producing the service name in provisioning system 2 generally reduces the
10 number to less than 40,000 by considering only subsets of features that directly affect parameters that must be sent to the switch. This is approximately 60 features. These 40,000 combinations of these approximately 60 features may be entered into a data table and define unique identifiers referred to herein as services names. Much of the data used in this embodiment for provisioning varies with the service names which are independent of switch type.

15 It should be noted that orders to initialize or change a service generally require consideration of the features that determine switch parameters. In this embodiment, approximately 60 features are considered when determining generic switch parameters that are generic, or common, to all services of a particular switch type. Generally, a service, as described, is a collection of features to which a customer subscribes. Each switch generally

GTE Ref. No. 99-881

requires a subset of features to determine the generic switch parameters. The approximately 60 features may be considered switch-independent. When considered collectively, the 60 features form a switch-independent list. Additionally, when forming a switch command, each of the features should be considered to determine if additional feature-specific text is required to be sent
5 to a specific switch. The 40,000 combinations in this embodiment are stored in the service names table that is described in more detail in conjunction with Figure 20.

This routine invoked at line 224 uses the feature name table and various values communicated to the program as stored in the service order data structure to determine a service name. The details on how to produce the service name using the feature name table and the
10 service name table will be described in conjunction with other figures. As a result of the call to GET_SERVICE , a primary class of service name is returned. Subsequently, at line 226 a call is made to GET _METRO to read the calling plan table. Generally, the calling plan returned from this routine is a boolean value which indicates whether or not a calling plan exists for the particular subscriber. In this embodiment, it should be noted that calling plan refers to a service
15 wherein a subscriber may pay a higher flat rate for the ability to call outside a local area without paying for each call. At line 228, a call is made with an EXEC_RPC command in this embodiment to execute a stored procedure within the NEP database. This call executes a query against the Vidar switch table, 460 in Figure 19, described in more detail in following paragraphs.

GTE Ref. No. 99-881

It should be noted that in this embodiment, the EXEC_RPC is an action or command that executes a stored procedure which generally eliminates the need to hardcode data in programs in order to handle various service and feature combinations. A database is designed and included in this embodiment to support all variations of parameters and switch syntax and the EXEC_RPC is used to access information included in the database. The EXEC_RPC call at 228 formulates the SQL query based on the switch ID (encoded in the CLLI_CODE in Figure 15, 280) the calling plan, and the service name as returned by the call to GET_SERVICE on line 224.

The code executed at block 230 performs a test to determine whether or not the primary class code has been found. The block of code at line 232 perform concatenation commands to form the switch command and various parameters for the Vidar switch type. In this particular embodiment the switch command being formed is stored in and referred to as the %CMD variable. Similarly, the code indicated by block 234 in Figure 15 combines any features and particular values in the commands for the switch. At line 236, the command %CMD is sent to the front-end server 24 to be executed by the particular switch. At line 238, the send key command is equivalent to pressing an "enter" key, or equivalent, with the value of one indicating to actually enter the command on the switch. In this embodiment, the value of 1 indicates that the NEP is to pause one second after sending the command before proceeding to the next instruction. This is because the switch is much slower than the NEP and will not be able to process the command in less than a second. At line 240, the chain command chains to a program called VIDAR_CHECK. In this particular embodiment, the CHAIN command at line 240

GTE Ref. No. 99-881

performs a context switch to a process called VIDAR_CHECK to await a switch response.

Additionally, the process may include error handling to be performed if there is an error during command execution as performed by a switch.

Referring now to Figure 16, shown is a flowchart that includes method steps of an
5 embodiment for the routine GET SERVICE as included in Figure 15 at line 224. Generally, the
method uses the feature name table to obtain an index and a 200 character service name
identifier. Each particular feature and associated value as included in the service order data
structure are associated with a particular position in the 200 character long service name
identifier. For example, if feature X is defined and has a value, it accordingly has an associated
10 character position in the 200 character long service name identifier. The value of the character of
the service name identifier may be determined in accordance with the existence or nonexistence
of a feature as well as the value associated with that feature. This mapping is indicated in the
feature name table, as will be described in conjunction with Figure 17-20.

At step 250, a lookup is performed using the feature name table for each feature name in
15 the service order data structure. A position in the 200 character long service name identifier field
the value for that particular position in the feature is determined in accordance with the feature
name table that is described in other paragraphs in conjunction with Figure 17. At step 254, the
service name formulated by performing a lookup in the feature name table for each of the values
uses of an index into service name table to obtain a 20 character name associated with the

GTE Ref. No. 99-881

service. Generally, the 60 or so features used in conjunction with this service names table affect the generic or non-feature specific switch parameters, such as the primary class code and others.

5 Generally, this technique is applied to isolate and consolidate switch dependency. For example, those features associated with a particular switch may be conceptualized and condensed into a single table as will be described in conjunction with other data structures and tables. However, in this embodiment, the switch dependency is consolidated into the lower level tables of the hierarchy.

10 In this particular embodiment, service is provided to phone subscribers with more than 400 features. There exists specific switch parameters for all the different switch models and variations as well as for a specific feature associated with each of these switches. Of these over 400 features in this particular embodiment, there are approximately 60 that effect the service name. These features may be subsetted and isolated.

15 In this embodiment, the service name is used in performing the SQL query on the switch database. Use of the 60 or so features reduces the number of records in the database. Alternatively, to include a database based on all features, such as 400 or so, causes many additional records to be included in the database.

GTE Ref. No. 99-881

Referring back to Figure 15, the call to EXEC_RPC at line 228 is accessing data included in a table that is switch dependent. The three parameters passed in the call are: 280 corresponding to the switch identifier (referred to as CLLI_CODE in the example), 284 corresponding to the calling plan, and 286 corresponding to the service name. In combination, these three parameters serve as a key indexing into the table to output a service class code which, in this example, is included in the switch command as detailed in code block 232. It should be noted that the detail of the Vidar switch table, along with other switch tables for other switches, is described in conjunction with other figures in paragraphs that follow.

Generally, the EXEC_RPC command may be used to index into the various tables in accordance with switch type to retrieve information that is used in formulating a command for the particular switch.

Referring now to Figures 17 through 20, shown are the various tables used are the different components included in the provisioning system 22 of Figure 1. Generally, these tables define a data schema used in formulating switch commands for use in the system of Figure 1. It should be noted that other tables than those included in Figure 17-20 may also be used, but these are of particular importance in formulating the switch commands in this embodiment.

The office description table 300 includes a switch ID and a corresponding description. The switch ID is the key used in this table to obtain a description of a switch. Generally the

GTE Ref. No. 99-881

switch description table is not used in this embodiment to process service orders, but rather may be used, for example, by a user or administrator in displaying information for viewing regarding a particular switch identifier.

5 The master switch table 320 may be read by the SRP and the programs using the switch ID as the key to obtain data to be included in a service order as stored in the service order database. In this particular embodiment, the SRP retrieves, for a given switch identifier (switch ID), the time zone and group ID information and includes this in the data structure stored in the service order database. Fields included contain data about the specific switches.

10 The switch identifier table 330 uses an office and switch number as may be maintained from the input data structure 18 to obtain as output the various switch identifier. The SRP reads from this table to obtain a switch identifier (switch ID) and stores it in the data structure included in the service order database 54.

15 The long distance carrier suspend table 340 may be read by the programs as executed in accordance with the particular ASDLs. The switch identifier and indicator may be used as an index into the long distance carrier suspend table 340 to return information regarding account credit management text or normal suspend text. The text returned as an output of accessing this table is actually included in the command to a switch. The long distance carrier suspend table may include information for a switch command, for example, if someone hasn't paid their long

GTE Ref. No. 99-881

distance bill and designates what type of action to take with regard to long distance service for the particular telephone subscriber.

The voice mail server table 350 uses as an index the switch ID, the area code and the exchange to return the voice mail server number for a particular subscriber. The SRP and the
5 program may read the voice mail server table.

The ringing frequency table 360 includes commands to a switch that determines the electrical frequency for making a ringing sound on a subscriber's phone. The key into the ringing frequency table is the switch ID, an equipment identifier, and a frequency which returns text in the switch text field that the program might be included in the actual command sent to the
10 switch.

The feature name table 370 may be read by the SRP in the program. As previously described, the SRP may read this to obtain information regarding the feature value pairs. Additionally, one of the executed programs in formulating a switch command may access this in conjunction with the service name formulation.

15 The feature table 380 may be read by a program to determine what common information to be concatenated or included in a switch command. The switch ID and feature name are used as indices to return "text to add feature", "text to remove feature", and "feature type code" field

GTE Ref. No. 99-881

data as included in the feature table 380. Generally, the field "text to add feature", and "text to remove feature" are included in accordance with whether the command to be formed is to remove or add a feature. The feature type code contains additional information for each feature.

5 In this embodiment, the feature type code facilitates switch support where different features require different commands to add the features. "TYPE1", for example, gets one command, and "TYPE 2" get another command. This may be in addition to the text that provisions the feature itself.

10 The default feature table 390 is read by the SRP to include default feature information, for example, for a particular geographic area. Used as an index into the default feature table is the switch ID and the feature name. An output of the table is the service type which specified the default. For example, whether a particular geographic area receives *66 call blocking information may be specified in the default feature table 390.

15 Tables 400, 410, 420, 430 and 440 are used in the embodiment only by programs in formulating switch commands. In particular, this is due to the fact the switch dependencies have been embedded in the classification of the ASDLs as associated with various programs that formulate switch specific commands. The 5ESS switch table 400 may be used to provide the switch parameters to initialize or modify a service. It may also be used to obtain the suspend service text information which is actually included in the command to the 5ESS switch. The

GTE Ref. No. 99-881

remaining fields in the table may be used as indices or keys to return the suspend service text
408.

The 5ESS switch by telephone number table 410 supports one switch in which the switch
parameters depend not only on the key fields defined for 400, but also on the subscriber's
5 telephone number. The 5ESS switch by telephone number table 410 takes as a key those fields
indicated by 412, and returns the remaining fields.

The DMS 100 switch table 420 uses as an index or key those fields indicated by 422.
The remaining fields of table 420 are returned as an output. A portion of these output fields, all
except the Centranet service indicator field, may be included in the commands formed for the
10 DMS 100 switch.

The GTD5 switch table 430 uses as a key field those indicated by 432, and returns the
remaining information some of which may be used in sending commands to the switch. It should
be noted that all of the non-key data included in 430 may be included in a command, such as an
initialize command. Other types of commands, such as a modify or restore command type, may
15 not include all the non-key data in forming a switch command.

The DCO switch table 440 uses as a key the fields indicated by 442 and returns the
service class code. The service class code may be included as part of a switch command.

GTE Ref. No. 99-881

Referring now to Figure 19, the DMS 10 switch table 450 is used to formulate commands for the DMS10 switch. The key is used as an index into the table as indicated by the field 452 and returns the service class code. It should be noted that all of the tables indicated in Figure 19 are read only by the programs to determine switch information and particular text that may be included in commands for the various switch types in this particular embodiment.

The Vidar switch table 460 uses as an index or key those fields indicated by the key 462 and similarly returns the service class code particular to the Vidar switch. Table 470 similarly uses the key as indicated by field 472 for the EWSD switch and returns information that may also be included in commands for the EWSD switch.

The calling plan table 480 is used to obtain calling plan information and is used also by the program in forming switch commands. An index into this calling plan table is with switch ID and exchange code. If a value exists in a calling plan table, a boolean value is returned of true indicating that the calling plan for this particular telephone subscriber exists.

The DMS 100 long distance carrier code table 490 uses a key which is indicated in field 492, and returns information, such as a translated long distance carrier code, which may be used in formulating switch command text.

GTE Ref. No. 99-881

The GTD5 hunt group table 500 uses as a key that which is indicated by field 502. Every field not included in the key field may be included in a switch command if it has a value.

The DMS 10 equipment prefix table 510 uses as a key that which is indicated by field 512 for the DMS 10 switch and returns a translated equipment prefix field which is used for formulating the switch command. This table may be used by a program in forming a switch command.

The DMS 10 error code table 520 uses as a key that which is indicated by field 514 and returns various error information.

Referring to Figure 20, the equipment translation table 530 may be used by the SRP to determine a particular user port in the specific switch identifier. It uses as an index those fields indicated by key 532 and returns a translated equipment identifier.

The GTD 5 traffic source table 540 is read by the program to generate report or statistics for a particular service. The key field 542 used as an index into this table. This table may be used, for example, to send a parameter to a switch to allow report generation from a switch database regarding switch traffic for routing, calls, trunk lines, and the like. The parameter may be determined in accordance with type of service and line equipment.

GTE Ref. No. 99-881

The GTD 5 pay-per-use feature table 550 is read by the program for a GTD 5 switch using the key as indicated as field 552. The remaining fields are returned in accordance with values specified for the key 552 and may be used in formulating commands sent to the GTD 5 switch.

5 The equipment identifier format table 560 is read by a program for a particular switch type and software version used as an index for key field 562 and an equipment identifier mask is returned. For example, in sending a command to a particular switch, this edit format as described by the equipment mask identifier may be used. Generally, this mask describes particular fields and values to be placed in them much like a date formatting structure for a
10 sequence of six digits to represent a month, day and year.

 The service names table 570 is read by the program which, as previously described, may be used to determine a service name that is used as an index into many of the switching tables also previously described. Recall that the service name identifier 572 is used as a key into this table. In this embodiment it is a 200-character field with particular positional encoded data in
15 accordance with features and returns a service name that may be from 1 to 20 characters in length. Other embodiments may have different character lengths.

GTE Ref. No. 99-881

The order type table 580 is read by the SRP and stored in the data base to determine the CSDL or service order type. The key field 582 indicates those fields which are used to determine the particular order type.

5 It should be noted in this particular embodiment that there are approximately 300 programs to implement over a million different types of feature commands and switches in combination.

10 In this particular embodiment, it should be noted that a restriction is encountered when referring to the mechanism of using parameters as may be seen in conjunction with Figure 12. Recall that the mechanism used is the dynamic setting of a parameter in conjunction with the processing described with Figure 12. However, in this particular embodiment, only a single parameter may be used. One problem encountered was this restriction of only being to communicate one piece of information with a single parameter. The problem arose with how to communicate other forms of information which may be needed to be communicated for the various commands.

15 One technique used to work around this restriction relates to dynamically modifying a data structure stored in the service order database 54 as used by the task coordinator in the NEP process. Generally, the data structure as used by the various processes may be dynamically expanded to include additional fields to communicate the information as needed. In other words,

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

a first portion of the data structure is used to store the service order information as may be included in the service order database 54 and used in performing the processing thereof.

Following this first portion of information is a second portion which is used as a parameter passing mechanism to communicate information between the various programs in the provisioning system 22.

Referring now to Figure 21, shown is an embodiment of a data structure with reference to the previously described data structure 100 which is used within the provisioning system 22 as initially produced and filled in by the SRP and stored in the service order database 54. It should be noted that the data structure 660 contains two portions, the original data structure 100 as created by the SRP and used by the task coordinator. Additionally, a second portion 602 may be added as a variable length portion in accordance with the mirror of parameters and values that need to be communicated to different programs in accordance with the different CSDLs and ASDLs. Generally, the technique is to create a new field in the data structure as needed with the appropriate values to communicate information dynamically. The self-modifying data structure format involves using the record for each data structure as a common area for data communication. This record may be written and read as needed from the various databases using, for example, database commands previously used to access other data. The knowledge of how to access the data and what is included is embedded within the programs. It should be noted that the common area or the additional segment does not need to be restricted to being appended to the existing data structure 100. Rather, it may be included in a beginning or other middle

GTE Ref. No. 99-881

portion of the data structure with knowledge of this location included in the programs that access the data structure and the additional parameters specified in this dynamic communication area.

The technique previously described minimizes the impact of changes to the code in accordance with the various number of switches. Additionally, multiple levels or hierarchy promote code reuse and modular code design as well as minimizing the impact on the code due to changes in switches. This multiple level hierarchy as previously described is used as a technique for determining what information from tables is needed to formulate commands for the various switches that may be included in the public switch telephone network. Additionally, in this particular embodiment, a self-modifying data structure is used as a common area for communication to communicate pieces of information and the values for them between various programs that actually formulate the commands sent to the switches.

It should be noted that any particularities in this system suggest a number of processes with a specific number of fields or tables or particulars to this embodiment. However, this should not be construed as a limitation to his particular invention as one skilled in the art may readily extend the principles of the invention included herein to other embodiments in other systems.

The use of a system commercially available such as the ASAP™ application by Architel™ may have analogous counterparts that provide the same function in other

GTE Ref. No. 99-881

embodiments of the invention in accordance with a particular implementation. It should also be noted in this particular embodiment that the database, such as the service order data base and the NEP data base of the provisioning system 22 may be SQL databases or other types of data bases. In this particular embodiment, these data bases are relational data bases that include, for example,
5 approximately 300 short programs as well as the various data tables.

In this particular embodiment, there is a set of programs particular for each switch, as well as a set of generic or common programs that function for all switch types and libraries of functions to provide for reusability of code. This greatly simplifies maintenance of code with regard to any changes being minimized to a particular module for a variety of switches.

10 In this particular embodiment, there were seven different switch types as well as tens of thousands of different services. Another embodiment may have a different number of switch types as well as a different number of services.

The general database schema and the use of tables with the information provided in the work order and the data structures provide a flexible and efficient mechanism for generating
15 commands in this provisioning system. In this particular embodiment, there are different tables for each switch type as well as some tables that are independent of switch type. The data schema used in this embodiment may be extended to other related fields. Generally, a system that requires generation and delivery of textual commands to a computer based system may be

GTE Ref. No. 99-881

accomplished using the techniques described herein with any extension that are recognizable by those skilled in the art. For example, incorporating the principles of the invention described herein, a system for provisioning commands for an electronic banking transaction may be used.

The foregoing description provides techniques for generating commands, as by the
5 second provisioning system, to replace the first provisioning system as included in the system 10 of Figure 1. When implementing a new provisioning system, such as provisioning system 2, the commands generated may be similar but not identical to commands generated by its predecessor systems, such as provisioning system 1. It may be desired to test the quality of the commands generated by the second new provisioning system prior that will replace the existing provisioning
10 system such as provisioning system 1 in the system 10 of Figure 1.

Many switches, as included in the PSTN of Figure 1, may be configured uniquely and success may be determined by comparing the output of the new system with that of the old system. A command comparison of the two systems for verification should allow for variations in word order and the use of optional phrases, such that a straight forward textual string
15 comparison is not necessarily useful in providing and verifying the commands generated by the second provisioning system.

What will be described in paragraphs that follow is a technique for verifying the commands generated by the second provisioning system in accordance with commands

GTE Ref. No. 99-881

generated by the existing first provisioning system. What is described below is a technique that reviews log files in accordance with commands generated by the first or existing provisioning system, and compares the commands included in the log files with the contents of database tables and program code comprising the new provisioning system, provisioning system 2, insuring that the necessary phrases, and the like exist in the new second provisioning system which generates the commands. Generally, this technique which will be described identifies words and phrases missing from the data sources used in command generation that are included in the second provisioning system. Thus, this indicates commands or options that the new system is not capable of constructing. It is anticipated that missing words in the new system generally result in failed command generation with various options, words or phrases that are missing. This technique therefore may be used to identify data omissions with regard to command generation. If no such problems are found regarding data omissions from data sources included in the second provisioning system, the second provisioning system may be expected to have the correct data to construct the commands for the particular switches.

Referring now to Figure 22, shown is an example of an embodiment of the system for analyzing and verifying the commands generated by the second provisioning system in accordance with commands generated by a first existing provisioning system. Included in Figure 22 is the first or existing provisioning system 20 as previously included in the system 10 of Figure 1. This existing provisioning system 1 generates one or more log files 700 which includes commands generated by the provisioning system 1 as used for switch commands to the

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

PSTN 26 in a system 10 of Figure 1. The analysis program 702 takes as input the various log files 700 as well as various data tables and programs from the NEP database 62 used by the second provisioning system in command generation. The analysis program 702 compares the various commands from the log files 700 with commands and options included in the tables of the NEP database 62 to determine if the second provisioning system is capable of correctly generating commands that are currently generated by the first provisioning system. In other words, the analysis program examines those commands actually generated by the first provisioning system. The analysis program subsequently determines which of these commands that are actually generated by the first provisioning system may be generated correctly by the second provisioning system 22 in accordance with tables used by the second provisioning system as stored in the NEP database 62. As a result, the analysis program 702 outputs analysis results 704 indicating, for example, what commands and options may not be correctly generated by the second provisioning system in accordance with commands output by the first provisioning system and data tables actually used in the command generation by the second provisioning system.

It should be noted that in this particular embodiment of Figure 22, there is an analysis program for each particular switch or hardware type. For example, there is an analysis program for the 5ESS switch type and the DMS10 switch type. Additionally, the log files 700 include commands that are sorted in accordance with switch or hardware type. In other words, the log files 700 includes a file or arrangement of 5ESS switch commands grouped together. Similarly,

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

the DMS10 commands are grouped together. Each of the hardware or switch type commands are read in as input to the switch type specific analysis program 702. In other words, the various commands associated with the 5ESS switch type as stored in the log file 700 are read as input into the analysis program 702 for the 5ESS switch type. Thus, the analysis in this particular embodiment is performed on presorted commands by switch type.

Other embodiments, as known to those skilled in the art, may use different techniques rather than presorting commands in accordance with switch or hardware type. For example, as will be described in paragraphs that follow in conjunction with the method of Figure 23, an embodiment may choose to do the sorting within the method steps as included in Figure 23. This is done as an alternative rather than performing the steps of sorting the commands prior to performing the method steps of Figure 23 as in this embodiment.

It should be noted that the log files 700 may be stored in an existing database format or as a file in a file system in accordance with the particular embodiment and software on a particular system. The log files may be stored on a first provisioning system, on a separate system, or on the same system upon which the analysis program is run. The various hardware and software configurations in terms of which of the components in Figure 22 were on a particular hardware and software system vary in accordance with each particular embodiment of the invention.

GTE Ref. No. 99-881

In this particular embodiment, the analysis program 702 is switch type specific due to the fact that in this particular embodiment hard coded portions of programs as included in the NEP database 62 are switch type specific. In other words, the analysis program in performing its analysis reads in various programs which include switch type specific data from the NEP database. In this particular embodiment, the analysis program or programs 702 may be written in the Perl programming language. This may be found on the Internet at the address using URL <http://www.perl.com>. Generally, Perl runs on a variety of commercially available processors with a variety of commercially available operating systems such as Windows 95™ and HP-UX™ and well as OpenVMS™. It should be noted that other embodiments may implement the analysis program using similar functions in other languages besides Perl. Some of these language processors may be commercially available or free on the Internet, similar to the way Perl is available on the Internet.

Referring now to Figure 23, shown is a flowchart of a method on an embodiment for analyzing commands generated by a provisioning system. At step 710, the log file for the particular switch type is scanned, identifying various switch identifiers to be analyzed. At step 710, a list of switch identifiers may be produced, identifying collectively a list of all of the switches identifiers which are of a particular switch type as being analyzed by the analysis program. Recall that a switch identifier identifies a unique physical switch in the network, such as the PSTN 26 of Figure 1. Various switch identifiers included in the PSTN may be of a particular switch type, such as the 5ESS switch type which is being analyzed by the particular

GTE Ref. No. 99-881

analysis program 702 for the switch type 5ESS. As a result of step 710, a list of all of the switch identifiers included in the log file for that particular switch type is produced.

At step 712, the data files, programs and the like as stored in the NEP database 62 are read in and stored in hash tables in accordance with the switch identifiers from the log file as identified in step 710. In one embodiment, these hash tables are stored in memory for speed in data storage and retrieval. Other embodiments may store a portion of these on disk files as well as in-memory copies.

As a result of step 712, data which is not associated with a switch identifier having commands generated in the log file do not have data read into hash tables and stored by the analysis program. This may be considered as an alternative in one embodiment rather than just reading in all of the commands for all of the switch identifiers from the log file for a particular type. In this embodiment, this results often in a shortened search time due to the fact that less data is stored in the various hash tables for data retrieval and storage. Additionally, this minimizes data storage of the various command options by only storing the data which is necessary for the various switch identifiers actually used identified in step 710.

At step 714, the log file records are accessed from the beginning of the file. A first record from the log file for this particular switch type is read in. At step 716, a test is performed to determine whether this is the last record or the end of the log file. If at step 716, a determination

GTE Ref. No. 99-881

is made that this is the end of the log file, processing stops and control proceeds to step 718. If it has been determined that this is not the end of the log file, control proceeds to step 720 where the hash tables are populated using the data from the NEP database as used in command generation.

These hash tables include known words and phrases as may be included in switch commands

5 generated and appearing in the log record. In contrast, words, parameters and the like not included in these hash tables may be considered unknown words or phrases. In this embodiment, for example, the known words and parameters as identified in accordance with hash table contents may be replaced with blanks, commas (","), or null characters. At step 722, a test is performed to see if the record from the log file is now empty. The logic of a test at step 722 is
10 that if all of the words and phrases for a particular command have been removed along with the command, then the record itself should be empty. Thus, at step 722, if the record is empty, all of the commands and various options for that command have been recognized and the provisioning system 2 may generate the command properly using its tables. Control then proceeds to the top of the loop formed by step 714 where another record is read into the log file for continued
15 processing.

At step 722, if the record is not empty, then at least one or more of the phrases and words from a particular record were not recognized or removed in accordance with the tables used at step 720. In other words, the tables read in from the NEP database 62 as used by the new provisioning system to generate commands did not contain all of the options or the command
20 words and phrases as generated by the old or existing system. Thus, it may be concluded that the

GTE Ref. No. 99-881

second provisioning system using these tables would not be able to generate an equivalent command. Thus, control proceeds to step 724 where error processing may be performed. Error processing may include, for example, generating messages about the unidentified words or phrases in a reporting file. Other types of error processing may be performed in accordance with each particular embodiment. Control proceeds to the top of the loop formed by step 714 where a subsequent log file record is read in and processing continues until the entire log file for that particular switch type has been processed.

Referring now to Figure 24, shown is an example of a flowchart of another embodiment while verifying the commands generated by a provisioning system. Generally, the method steps of Figure 24 are similar to those of Figure 23 with additional steps regarding what actions are taken when commands are recognized, as well as what information is output, for example, in an analysis report.

At step 730, the log file is scanned for switch identifiers to be analyzed. This step 730 is analogous to step 710 of Figure 23 in which a list of switch identifiers for this particular switch type is output. At step 732, data files, programs and the like are read into hash tables in accordance with the various switch identifiers from the log file. Step 732 is similar to step 712 of Figure 23. At step 734, the first record of a log file is read. Step 734 is similar to step 714 of Figure 23. At step 736, a determination is made as to whether the log file record just read in identifies a new switch identifier. At step 736, if a determination is made that this is a new

GTE Ref. No. 99-881

switch identifier, control proceeds to step 738 where summary information is printed out for a previous switch identifier if there is any.

At step 740, the existence of command switches in hash tables is determined for the new switch identifier just read in in step 734. Generally, step 738 prints out summary information for the previous switch identifier, if there has been a previous switch identifier. At step 740, various data tables, programs and like as read in from the NEP database 62 and stored in the hash tables in step 732 are scanned to determine if in this particular new switch identifier there is information stored in the various tables. For example, for a particular switch identifier there may be no existing entry in a particular data table. Following step 740, control proceeds to the top of the loop at step 734 where another log file record is input.

In this particular example, it is assumed that there is a record delimiter when a new switch identifier and its commands are encountered. Thus, the first time a new switch identifier is encountered, there is a separate record identifying this changeover to additional commands for this particular new switch identifier.

If at step 736, a determination has been made that the switch identifier for a particular record read is not a new switch identifier, control proceeds to step 742 where a test is performed to determine if we are at the end of the file. If at step 742 it is determined that we are at the end

GTE Ref. No. 99-881

of the log file, control proceeds to step 748 where summary information is printed for the current switch identifier. Subsequently, processing stops at step 750.

At step 742, if a determination is made that it is not the end of the file, control proceeds to step 744 where known words and phrases from the command are removed in accordance with the hash tables as stored at step 732. At step 746, any unidentified remaining text is output along with the original command for the purposes of error processing in step 746. Subsequently, control proceeds to step 734 where processing continues with the next file record.

Referring now to Figures 25 through 28, shown is an example of the output as produced in the analysis results 704 Figure 22 in one embodiment. It should be noted that what is described and shown in these figures and in paragraphs that follow is an example of an embodiment of the output of the analysis that may be performed by the analysis program 702. Other embodiments may employ different analysis results 704 in accordance with the principles of the invention described herein.

In Figure 25, the report identifies a particular switch identifier 760 as LNLKFLXA99H. At lines 762, information regarding the existence of information in tables regarding this particular switch identifier is output. Similarly, lines 764, 766, 768, and 770 are results of performing, for example, the processing of step 740 of Figure 24 where the existence of the various command switches and the hash tables for the switch identifier is determined.

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

In this particular analysis report embodiment, an indication is output as to whether the various tables include entries for this particular switch identifier. Additionally, actual information as stored in the table for this particular switch identifier is output, for example, at lines 764.

5 Beginning with output at line 776, the various unidentified text, words and phrases are output. For example, 772 identifies a line of text that has not been identified for a corresponding command line 776. This is indicated by having the analysis program for the switch type 702 output the unidentified text 772, followed by the original command 776 which includes the unidentified portion within that command 774. Similarly, the original command as identified by
10 780 is output. The unidentified text portion 778 is identified within the command 780 in field 782.

Referring now to Figure 26, after the switch identifier specific information is output, summary information may be output regarding the current switch identifier or identifiers, for example, the lines identified by 786. Information is output regarding switch types as identified
15 by 784. In 786, the unidentified words for the records associated with identifiers 784 are output.

Referring now to Figure 27, additional summary information regarding the two switches, switch identifiers 784, are output beginning at line 788. For example, beginning at line 788, switch features table for the two switch identifiers 784 are output.

GTE Ref. No. 99-881

It should be noted that some of the files and tables read in from the NEP database 62 as used in the processing step 712 of Figure 23 and step 732 of Figure 24, for example, include switch independent, as well as switch dependent information. In other words, as previously described, some of the tables contain switch independent information and other tables contain switch dependent information in accordance with switch type. For this reason, in this particular embodiment some of the programs, for example, are switch dependent. Thus, it may be a design decision to perform the analysis based on hardware switch type. Other variations of this embodiment are possible in accordance with each particular embodiment's specifications incorporating the principles of the invention as described herein.

Generally, what has been described is a method for verifying the commands generated by a system in accordance with a log file of an existing systems command. A log file 700 of Figure 22 includes successful switch commands which are needed to compare against the phrases available and generated by a new provisioning system, such as provisioning system 2. Various files as included in the NEP database 62 containing words and phrases stored in database tables and embedded or hard coded into programs themselves are used in this second provisioning system. The method described in conjunction with Figures 23 and 24 include reading the switch command work file to identify switches to be reviewed. Subsequently, data is stored in the NEP database and related program files are loaded into hash tables with various switch identifiers having commands generated by the first provisioning system. Finally, the log file is again reviewed one record or command at a time. For each command, generally the methods and

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

techniques previously described examine words and phrases and compare those against the available database and program words and phrases. If a match is found, it is removed from the command. This continues until either there is nothing left to remove from the command or no other words or phrases are left to be compared. If there is nothing left to remove, this is a
5 successful test of the system. The new provisioning system may be concluded to contain sufficient words and phrases as stored in the tables in the database to construct the command. Any words or phrases left over are printed, as indicated by the second or alternate method of Figure 24, followed by the original command for review by a person. Generally, these items which are printed as left over need to be added to the system to enable a second or new
10 provisioning system to create the command with the various options which are missing. Additionally, the second method of the steps of Figure 24 output additional information for review by a person, for example, such as outputting the contents of the various databases for review by an engineer, and a summary of the words or phrases missing from the data tables for a particular switch identifier.

15 Generally, the testing techniques described herein for verifying the commands generated by the second provisioning system may be used to verify other types of command or automatic generation system. Generally, the techniques do not focus on coverage of commands that may be generated by the first or existing provisioning system. Rather, the focus is on the actual set of commands generated by the first provisioning system. This approach focuses on actual rather
20 than potential command generation. One advantage of the foregoing technique, for example, is

GTE Ref. No. 99-881

that commands which may be outdated or old in an existing system which are no longer used or generated are not considered in the new provisioning system. For example, if code exists within the first provisioning system to generate outdated commands, and no one ever uses those code segments from the old provisioning system, this may be considered "dead code". Commands
5 generated by this portion of dead code do not need to be considered during the command generation of the new provisioning system, such as provisioning system 2. By examining the actual commands output and included in the log files, only those commands actually used are generated by a provisioning system are considered.

The foregoing techniques also verify the correctness of a command in a manner which
10 does not check for actual syntax and positional dependency within a command, for example, as generated by the new provisioning system. Rather, a given command actually generated by an existing provisioning system, the various tables are checked to determine if the second provisioning system has the capability of generating the various words or phrases included in an actual command. This allows for the second or new provisioning system to generate commands
15 in different orders or with a variety of syntax which are equivalent in terms of function to commands generated by the first or existing provisioning system. This eliminates one of the previously identified problems, for example, where a command generated by the first provisioning system may have different qualifiers or parameters in a different order, or where a series of commands produce equivalent function for a single command, for example, generated
20 by the existing provisioning system. This is used as a technique for determining command or

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

functional equivalence rather than performing a syntax or a string comparison, for example, to verify whether commands match between the two systems. The foregoing technique of allowing the variety of different switches and commands in different orders implies that this verification technique allows for verification of handling different dialects as generated by the second provisioning system. Commands may not be identical, but they may in effect produce the same correct results in generating commands. Generally, the foregoing techniques provide for quality verification of commands by an existing system by focusing on command equivalents in terms of functions, and equivalent vocabulary existing in tables of a new system.

When implementing a new provisioning system, such as provisioning system 2 of Figure 1, it may be necessary to determine the various switch specific command options as generated by an existing provisioning system in order to populate those data bases and tables included in the second new provisioning system. In other words, it must be determined what switch identifier specific command options are currently generated and used by a first provisioning system in order to populate the data base and tables of the second new provisioning system to generate switch specific commands.

What will be described in paragraphs that follow is a technique for automatically extracting commands and command options from a first provisioning system to be included in the database of the second new provisioning system to allow for switch specific command and command option generation in an automated fashion. Generally, this technique which will be

GTE Ref. No. 99-881

described determines an actual working set of switch commands and switch command options.

Accordingly, it is determined what outputs correspond from the first provisioning system for a particular working set of command and command options as an input. In other words, rather

than figure out all possible command and command options which may be generated by an

5 existing provisioning system, a technique is used whereby only those command and command options actually in use by current telephone subscribers are considered. Thus, any dead code

which may be included the first existing provisioning system which is no longer used to actively generate command and command options is not considered in the new provisioning system by

generating a simulated set of telephone subscription service orders which correspond to this

10 actual working set and in putting this into the existing provisioning system, a controlled set of

output is obtained. This controlled set of output is correlated with corresponding order input and

the various command and command options for each specific switch identifier and order are used

to populate various data tables included in the second provisioning system 2, 22 of Figure 1.

It should be noted that those techniques described herein may be applied by those of

15 ordinary skill in the art to other types of systems and applications which generate commands and command options, such as electronic banking systems.

Referring now to Figure 29, shown is an example of the embodiment of a block diagram of a system used to automatically extract and generate switch specific command and command options which may be included in the second provisioning system to populate the data tables in

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

accordance with various commands generated by an existing first provisioning system. System 801 includes components similar to that of system 10 of Figure 1. In particular, customer contact data 12 is input to an order inventory system 14 which uses information from a telephone subscriber data base 16. The order inventory system 14 generates an order data structure
5 corresponding to the subscriber order of the customer contact data 12. This order data structure is input to an existing first provisioning system 1, 20 which outputs various switch specific commands to the front end server 24. The front end server 24 sends the switch specific commands to a designated switch included in the PSTN 26. Each of these components are also included in the system 10 of Figure 1.

10 Also included in the system 801 of Figure 29 is an order generator 802. The order generator 802 generates a set of simulated orders which act as an input to the order and inventory system 14. Generally, the order generator 802 inputs a fixed format order similar to that which is input by the customer contact data 12 to the order and inventory system 14. Additionally, the order generator stores various pieces of order specific information into an order identifier table
15 800. This order identifier table stores information based on various service orders and input into the order inventory system 14 for which a corresponding switch specific command is generated by the first provisioning system 1, 20 sent to the front end server 24. The order filter and command extractor 804 takes as input the command generated by the provisioning system 1, 20. The order filter and command extractor 804 uses the switch specific command generated by a
20 provisioning system and information stored in the order identifier table 800 to correlate, for a

GTE Ref. No. 99-881

given input set, what switch specific output parameter command option or options are generated by the first provisioning system 1, 20. In turn, the order filter and command extractor 804 uses these various command options as generated by an existing provisioning system 1, 20 to populate data tables included in the data base of the second provisioning system 2, 22.

5 The technique used in the system of Figure 29 is an alternative to determining the complete set of commands and switch specific command options as may be generated by the provisioning system 1, element 20, of Figure 29. In other words, a working set of switch specific commands and command options is determined and automatically generated by the order generator program 802. Thus, this system tests or simulates various service orders generation for
10 those which may actually be generated by the provisioning system 1 based on switches and telephone subscriber options in use. The order generator 802 generates a corresponding order as input into the order and inventory system 14.

 The order generator 802 generates service orders for each switch identifier. In other words, for each switch identifier, every combination or features actually in use by at least one
15 customer or telephone subscriber is generated. Thus, the set of generated or simulated output by the order generator 802 represents the working services currently in use for each particular switch identifier. It should be noted that the order generator may obtain the various features actually used by the customers from an existing data base or other list of customer and subscriber service information 803. It should be noted that other embodiments may determine the actual working

GTE Ref. No. 99-881

set of subscription data using other techniques in accordance with each particular application and embodiment.

Referring now to Figures 30 and 31, shown are flowcharts of an example of an embodiment of the method steps for automatically extracting switch specific command options from a first provisioning system as used to populate databases and data tables included in the second provisioning system. At step 820, for each switch identifier, a simulated order set is generated. As previously described, a switch identifier identifies a specific switch as may be included in the PSTN 26 of Figure 29 in the system 10 of Figure 1. For each switch identifier, an order set is generated which corresponds to a working set of services for the various subscribers for that specific switch identifier. In this embodiment, in other words, every possible combination actually used by at least one customer subscriber for that particular switch identifier is determined. Accordingly, an order set of the various features and services actually in use for a particular switch identifier is determined.

Each of these orders generated as a result of step 820 may be in the format of a particular phone number as associated with a class of service which represents a combination of the various features in use or that may be used by a telephone subscriber. At step 822, the various orders are sorted by class of service or various combinations of features. In other words, at step 822, common classes of service or features are grouped together for each switch identifier. Generally, as a result, duplicate sets of features are identified and grouped together.

GTE Ref. No. 99-881

At step 824, feature numbers are translated to feature names. In other words, it is assumed that the data as included at step 822 includes numeric components which represent the various features or combinations of features available to the different telephone subscribers. At step 824, an alphabetic or alphanumeric equivalent for each of the numeric codes is determined as a result of step 824. The various tables as may be used to determine and translate the feature numbers to feature names are based on this particular instant and that customer contact dated 12 is typically first input into some kind of numeric field which is translated upon input to the ordered inventory system 14 to an alphabetic or alphanumeric equivalent. Thus, to simulate the same input from the order generator program and put to the order inventory system, an equivalent translation must be made by the order generator program 802. In this particular instance, the feature names actually generated by step 824 may be those actually included in the final fixed format record which is output by the order generator 802 and access input to the order and inventory system 14 representing the various subscription orders. It should be noted that in this embodiment, those feature names translated at step 824 are actually included in the feature name table as included in the second provisioning system.

At step 826, those features not used in determining the class of service are eliminated from the various subscription orders. In other words, as previously described, approximately 60 of over 400 features in this particular embodiment are used in determining class of service. Thus, any features included in a particular service order which are not actually used in determining a class of service, or not actually included in this subset of approximately 60

GTE Ref. No. 99-881

features, are filtered and eliminated as a result of step 826 from each of the generated service orders.

At step 828, the order identifier table 800 as included in Figure 29 is updated with these feature names for each corresponding service order identifier. At step 826, a service order
5 identifier is determined which corresponds to the various combination of features representing a particular class of service for this service order. At step 828, the order identifier table is updated with an order identifier and the corresponding feature names representing that service order for that combination of features. At step 830, the service name for each group of feature names stored in the order identifier table is determined. At step 832, the service names are also added to
10 the order identifier table 800. Thus, as a result of the processing to step 832, the order identifier table is populated with an order identifier or surface order ID as previously described with a corresponding service name and the various combinations of corresponding features for that particular service name.

In this embodiment, the service name may be determined by using various tables and data
15 structures as may be included in the second provisioning system 22. Recall that the feature name table 370 of Figure 17 may be used in conjunction with the service names table 570 to determine the service name corresponding to a particular class of service or combination of approximately 60 features which have been subsetted. Recall that each feature name is represented in a vector representation by a corresponding position in a service identifier. In other words, the feature

GTE Ref. No. 99-881

name table 370 as previously described for particular feature name has a position, such as a column within the service identifier. The feature name table 370 determines which position in the service identifier corresponds to a particular feature name. By examining at a combination of various features using the feature name table, a vector of values may be determined representing a combination of the various features for a particular subscription order. This forms a service identifier which may be used as an index into the previously described service names table using, as a key, the service name identifier 572. Accordingly, a service name may be determined which represents or corresponds to the various service name identifiers for a combination of features.

It should be noted that the tables used to determine the service names at step 830 may be those actually stored in the second provisioning system, or equivalent tables storing equivalent data in other locations. The actual location of the various tables used in steps 830 and 832 may be determined in accord to each particular embodiment.

Referring now to Figure 31, control proceeds from step 832 to step 840 in which formatted service order commands are generated and input to the order and inventory system 14 of Figure 29. In other words, the order generator program 802 updates the order identifier table with service name, order identifier, feature names and also outputs or generates formatted service order commands in accordance with a predefined format of the order and inventory system. It should be noted that in this particular embodiment, this is a fixed format order which is generated and expected by the order inventory system 14. Other embodiments may use different systems to

GTE Ref. No. 99-881

implement the other inventory system, and thus the actual form of the order generated by the order generator 802 may vary with each particular embodiment and implementation.

At step 842, switch commands are generated by provisioning system 1, 20. At step 844, the switch commands as generated by provisioning system 20 are intercepted and examined to
5 determine if the service order identifier matches one of the simulated order sets as generated by the order generator program 802. In this particular embodiment, referring back to Figure 29, the order filter and command extractor 804 takes as input the commands generated by provisioning system 1, 20 and uses information stored in the order identifier table 800 to determine if the service order identifier for a given command generated by the provisioning system 1 corresponds
10 to an entry in the order identifier table 800 as generated by the order generator 802. For each of these particular service orders matching one order identifier table 800, the order filter and command extractor extracts the parameters from the various switch commands at step 846. Using the information stored in the order identifier table 800, and the parameters and switch commands for each switch type as extracted from the commands generated by the provisioning
15 system, the order filter and command extractor 804 populates switch specific data tables as included in use in the second provisioning system 22, as in step 848.

For example, referring back to Figure 18, tables 400, 420, 435, 440, and 450 may be populated with switch specific command options. For example, referring to Figure 18, the 5ESS switch table 400 may be populated. The switch identifier may be obtained from the command

GTE Ref. No. 99-881

message generated by the provisioning system 1 to the front end server. Similarly, the service name and other parameters or records included in the 5ESS switch table for each record entering may be obtained from data included in the order identifier table 800, as each of the different fields included in a record will present the different features.

5 Referring now to Figure 32, shown is an example of an embodiment of the order identifier table 800 as may be included in the system 801 of Figure 29. The order identifier table 800 includes in this particular embodiment three fields for each record. A service order ID 102 has corresponding service name 810 and feature names 812. It should be noted that the service order ID 102 corresponds to the service order ID as included in the new data
10 structure 100 of Figure 6 previously described. The service order ID corresponds to various combinations of features for this particular service order. As determined by different data tables, the various combination of feature names corresponds to a given service name.

 Also as previously described, the order generator program 802 creates the order identifier table 800. The order filter and command extractor 804 uses the order identifier table 800 and
15 various switch specific commands and command options as generated by the provisioning system 20 to populate switch specific tables as may be included in the provisioning system 2.

 It should also be noted that various fields included in the switch specific tables, such as the DMS 100 switch table and the GTD5 switch table 430, each may include text actually used in

FHE Ref. No. GTM 6301

369948.3

GTE Ref. No. 99-881

command generation. It is this information, along with the various feature specific information, which is populated into the different switch specific data tables in accordance with commands generated by the provisioning system 20 and information stored in the order identifier table 800.

As an example, the GTD5 switch table 430, as previously described in conjunction with

5 Figure 18, includes data that may be included in a command. Thus, various fields of information as included in a record of the GTD5 switch table is actually text from a generated switch command from the provisioning system 20 stored in a record with the corresponding service name for the various servicing features as obtained from a corresponding order in the order identifier table 800.

10 It should be noted that the order generator 802 may be a single program or a combination of various programs executed on different machines or hardware processors. Generally, these one or more processors may be commercially available processors. The various functions as specified in this particular embodiment as performed by the order generator 802 may be implemented for use on one or more of a variety of commercially available processors using
15 commercially available programming languages and scripting languages. For example, one embodiment, referring back to the steps of Figure 30, sorts orders as a result of step 822 by using a scripting language included in a particular implementation of the order and inventory system 14. The steps of determining a service name for a particular combination of features as well as filtering and eliminating those features not used in determining the class of service or
20 service name may be performed on one of the variety of commercially available processors such

GTE Ref. No. 99-881

as an Intel-based processor. The filtering of those features, as performed at step 826, not included in the service name may be performed by a program written in the PERL programming language previously described. The determination of the service name in accordance with the various features may be written in a commercially available programming language, such as "C",
5 also executing on a commercially available processor running the UNIX operating system. The programs written in PERL and "C", respectively, for filtering and identifying a service name may both be executed on a different processor from software performing the functions of the previous processing steps, such as 820, 822 and 824. In particular, if the data tables which are used include using a service name and feature names table, the processing steps associated with
10 these functions may be performed on a different system having access to these data tables stored in the second provisioning system. Subsequent processing steps such as 840 may be performed on another system. The step of formatting service order commands with a particular fixed data format as step 840 may be performed by a commercially available Intel-based processor. The program which generates these fixed format service orders may be, for example, written in a
15 commercially available language such as Basic executed on an Intel-based processor.

Thus, as known to those skilled in the art, various techniques, programming languages, and the like may be used to implement the various steps described in conjunction with the flowchart of Figures 30 and 31. Also in accordance with each particular embodiment, data may be stored in a variety of different configurations in accordance with system hardware and
20 software configurations that perform the method steps of Figure 30 and 31. Accordingly, with

GTE Ref. No. 99-881

each particular embodiment, different programming languages and different processors may be used in combination to generate and implement the output and the functions of the method steps of flowchart of Figures 30 and 31.

5 Generally, the techniques described herein for generating and populating various switch specific tables used in a second new provisioning system are in accordance with commands generated with an existing first provisioning system. Generally, the techniques do not focus on the coverage of commands and options that may be generated by the first or existing provisioning system, but rather focus on the actual set of command and options actually generated by the first provisioning system in accordance with a working set of subscription
10 services and options actually in use by customers. This approach focuses on actual, rather than potential, switch command options. One advantage of the foregoing technique, for example, is that commands which may be outdated and no longer in use, and thus no longer generated by an existing system, are not included in the second newer provisioning system. This is similar to the concept previously described for testing. By first determining the actual set of subscription
15 services and features in use by customers today, only those commands actually generated by the existing provisioning system are considered when populating data tables used in a second provisioning system.

While the invention has been disclosed in connection with preferred embodiments shown and described in detail, their modifications and improvements thereon will become readily

GTE Ref. No. 99-881

apparent to those skilled in the art. Accordingly, the spirit and scope of the present invention is
should be limited only by the following claims.